Europäisches Patentamt | European Patent Office | Office européen des brevets

REC'D 05 NOV 2003

WIPO PCT

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

02292651.3

PCT /IB/03/4613

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

[illegible]

R C van Dijk

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Anmeldung Nr: Application no.: Demande no:	02292651.3	Anmeldetag: Date of filing: Date de dépôt:	24.10.02

Anmelder/Applicant(s)/Demandeur(s):

SCHLUMBERGER Systèmes
50, avenue Jean Jaurès
92120 Montrouge
FRANCE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Protection against DOS attacks on smartcards

In Anspruch genommene Prioriät(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06K19/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

Protection against DOS attacks on smartcards

1. State of the art

1.1. *Description*

One of the fundamental features of smartcard technology is the ability to act as a secure repository for various credentials (PINs[1], keys, unblock codes, etc.).

Currently, when credentials are verified by a smartcard for authentication purpose (i.e. when the smartcard tries to verify the identity of the entity requesting a service, such as a user, a terminal, a server, an administrator, or an application) an internal counter is decremented each time the verification fails. When this counter reaches zero, the associated credentials are blocked.

1.2. *Problem(s) raised by state of the art solutions*

1.2.1. Introduction

Some credentials can be unblocked or reprogrammed (by entities having sufficient privileges), but others can't, and blocking them often results in a need to physically replace the smartcard.

For example, let's consider the following scenarios (we assume that a file system smartcard is used, and the PIN is under the root directory of the smartcard):

- ➢ If a user blocks his PIN, an administrator can unblock it with the unblock code
- ➢ If both the PIN and the unblock code are blocked, a smartcard management system could repersonalize the smartcard using the transport key
- ➢ If only the transport key and unblock codes are blocked, then the card is still usable as long as the PIN is not forgotten/blocked by the user and no

[1] Personal Identification Number, which is a secret code typed by the user to authenticate himself

administrative operations are needed on the smartcard (loading of new file structures, etc.)

- ➢ But if the PIN, the unblock code and the transport key are blocked, you must change the smartcard physically

1.2.2. DOS attacks

Smartcards were not initially designed for use as security devices protecting personal computers (PCs) and networks interconnecting those PCs.

A family of attacks that is very relevant in such context (especially in corporate environments) has not been taken into account when specifying smartcards features.

One of the most popular and frequent attacks on the internet is the DOS (denial of service attack). This consists in attacking a component of the network (server, etc.), often by overloading it with requests. The component becomes unable to perform its duties, and as a consequence the end users are stuck. The system is not compromised, but it is not usable anymore.

If a virus hits a big organization and tries wrong credentials until smartcards get blocked (e.g. by sending *verify_key* commands with random data), thousands of users would be unable to work.

While end users' PCs can be recovered automatically thanks to backup systems, smartcards would need to be physically replaced. During this replacement phase (which might be very long and costly for big organizations) users would be unable to log on to their PCs and secure their network, or would have to do this with usernames and passwords which have a much lower security level than smartcards.

1.2.3. Key blocking at development stage

One of the most frequent calls to the technical support teams is due to smartcards that were accidentally blocked and cannot be programmed anymore.

The technical supports need to figure out that the cards were actually blocked (people often think that there's a problem with the smartcard or with the smartcard software development tools that they are using).
Then they have to supply the developers with a new set of cards.

2. Invention

2.1. Summary

The idea consists in modifying the way the credentials' counter is managed, by adding a second counter, which works slightly differently.
Once the first counter reaches zero, rather than blocking the credentials (as done currently), the smartcard starts decrementing the second counter.
The second counter is associated with a waiting loop mechanism.
Due to specific constraints of smartcard technology, the design of this waiting loop requires specific features that are part of the invention.
Examples of such constraints:
Smartcards

- ➢ have no permanent clock
- ➢ can be removed from the smartcard reader at any time by the user
- ➢ can be remotely reset by an attacker at any time
- ➢ have to comply with ISO 7816 standards
- ➢ etc.

2.2. Diagram

The diagram displayed on the next page shows how the anti DOS mechanism can be put in place on standard ISO 7816 compliant smartcards.

More details regarding the rules that should be followed when putting this mechanism in place are given in the following chapters.

N.B. The same idea could be implemented in other environments (e.g. USB tokens, etc.), in which case the rules might have to be adapted to the constraints of the new target.
However, the basic principles described herein still apply.

For the sake of clarity, the changes in the existing systems have been highlighted in red.
Some components are added, but none are removed.
Any component that is not red is currently existing in state of the art smartcards.

See figure 1.

2.3. Description

2.3.1. Attempts counters

As described in 2.1, the existing attempts counter is kept, and it is complemented with a second attempt counter.

The first counter is unchanged (usually such a counter has an initial value varying between 1 and 15). Most often, the counter is predecremented before each credentials verification. If the verification succeeds, the counter is reset to its maximum value, otherwise it is unchanged.

The second counter (the new one) starts being decremented only after the first counter reaches zero. When the first counter reaches zero, it is no more decremented, but the credentials are not yet blocked. Instead, the second counter is predecremented before each new credentials verification. If the presented value was good, both counters are reset to their respective maximum values, otherwise they are unchanged and a waiting loop is performed. If the second counter reaches zero, the credentials are blocked.

The initial value of the second counter is discussed in 2.3.7.

2.3.2. Waiting loop counter and waiting loop flag

A waiting loop counter and a waiting loop flag are managed in EEPROM (or in any available programmable non-volatile memory). Both have a global scope, i.e. they remain visible outside the context of the waiting operation and of the credentials verification.

The flag is initially cleared, while the initial value of the counter is discussed in 2.3.7.

There's only one waiting loop counter and one waiting loop flag for the smartcard, while the number of attempts counters is proportional to the number of credentials on the smartcard.

2.3.3. Waiting loop

The waiting loop consists in:

1. Setting the waiting loop flag
2. Spending a predefined amount of time, for example by performing a series of NOP (no operation, i.e. dummy instruction), not exceeding the maximum duration negotiated thanks to the ATR (Answer To Reset) since we don't want to time out
3. Informing the host that the smartcard is alive (i.e. it has not timed out). Following ISO 7816 constraints, this consist in sending a specific byte to the host (the value $60 to be more accurate)
4. Decrementing the waiting loop counter, and if it is non zero going back to step 2
5. Resetting the waiting loop counter to its maximum value
6. Clearing the waiting loop flag

Note: alternative means can be used instead of the waiting loop counter (e.g. if the chip has a timer available), but the principle of the waiting loop remains the same. In any case, a marker indicating the current level of completion of the loop needs to be stored in non volatile memory.

2.3.4. Resuming the waiting loop

The waiting loop can be interrupted, either accidentally or on purpose.
For example, the end user might wonder what's happening with his smartcard and remove it from the reader, or an attacker might want to block the credentials quicker and send a reset order to the smartcard in order to stop the waiting loop mechanism.

In order to prevent this, the waiting loop status (counter and flag) has a global scope (as explained in 2.3.2), and a component of the smartcard (the APDU manager) is modified in order to resume the waiting operation in case it was interrupted during a previous session.
This mechanism is described more in details in the next section (2.3.5).

2.3.5. APDU manager modification

Currently, when a smartcard is reset, it performs a certain number of operations (various tests, selection of the communication protocol, of the voltage for the power supply, of the communication speed, etc.) then if everything is OK it switches to a mode where it can receive orders from the host.
Those orders are called APDUs (application protocol data units).
The piece of software running on the smartcard and responsible for receiving APDUs from the host and dispatching them (either to the smartcard operating system, to the applets that have been loaded, or to any relevant module) is called the APDU manager.
The APDU manager has to be modified in the following manner.
Before an APDU is dispatched, the APDU manager has to check the state of the waiting loop flag.
In case the flag was cleared, everything works as before (i.e. the APDU is processed normally by the smartcard).

Otherwise, it means that a waiting loop has been interrupted and needs to be resumed.

The APDU manager calls the waiting loop mechanism described in 2.3.3.

Since the waiting loop counter is stored in non-volatile memory and has a global scope, the waiting loop continues where it had previously stopped.

In case the waiting loop is interrupted again, it will be recovered thanks to the same mechanism.

Only when the waiting loop has been completely performed will the APDU manager start processing the APDU that was called. From the external world, the smartcard will behave exactly as before except that the execution of the APDU will take much longer than it does normally.

Optionally, certain APDUs (such as diagnostic APDU) can be allowed prior to the waiting loop. C.F. 2.3.9 for more details.

The reason for performing the waiting loop in the first APDU that is sent to the smartcard is twofold.

- First, the waiting loop needs to be compliant with ISO constraints and must be transparent to the existing systems (no update on the client software should be mandatory in order to deploy the DOS protected cards). The waiting loop cannot happen at any time (if it's done just after reset, the smartcard might be considered as not working). Also, Windows 2000 and XP power down the smartcard when no connections are made during a certain time, which justifies informing the host that the smartcard is alive and processing (thanks to the procedure described above).
- Second, this waiting loop serves as a protection avoiding that the credentials be blocked, but it also has to warn the user that an attack (or a bug, in case it happens at development stage) is threatening the smartcard. So in order to be noticed, the waiting loop must occur at a time when the smartcard is expected to perform certain operations and return a result.

2.3.6. Types of credentials for which the invention is relevant

The invention is potentially applicable to any kinds of credentials, however it is more secure and useful on certain than on others. For this reason the parameters tuning discussed in 2.3.7 will be adapted to the type of credentials you deal with. A more detailed discussion on the security impact of the invention depending on the type of credentials can be found in 2.3.8.

2.3.7. Parameters tuning

Two parameters need to be fine-tuned:
The duration of the loop and the maximum number of slowed attempts.
The first one is proportional to the waiting loop counter (and is unique for the whole smartcard), while the second is directly linked to the new number of attempts counter introduced in this invention (and can be different for every credentials).
Several conflicting constraints determine the best value for those parameters. Some of those are listed below:

- ➢ The maximum number of slowed attempts multiplied by the duration of the loop should be long enough to render the DOS attack success very unlikely
- ➢ The maximum number of slowed attempts should be small enough to not increase the likelihood of credentials guessing attacks (see chapter 2.3.8)
- ➢ The duration of the loop should be long enough for the user to notice that something is going wrong and report it to the helpdesk
- ➢ The duration of the loop should be short enough for the users not to be blocked too long during their work. Indeed, although this state is temporary (and does not require any intervention on the smartcard in order to come back in a normal state), it is inconvenient
- ➢ Etc.

As an example, a waiting loop of approximately 30 (thirty) minutes and a maximum number of 100 (one hundred) slowed attempts seem to be

reasonable parameters for a transport key, for open platform keys, and for unblock codes.
Assumption: the above keys and codes need to be strong (i.e. chosen randomly or cryptographically, by a diversification mechanism etc.).
For PIN codes, the maximum number of attempts should be much lower, for example 5 (five), unless a very robust PIN policy has been defined and enforced, C.F. 2.3.8. This increases the probability (which remains extremely unlikely) of a successful DOS attack on the PIN, but such attack could be recovered without changing the smartcard physically and does not represent a significant threat.

Of course the actual values can be customized at personalization stage according to the exact application and security requirements.
The smartcard OS should prevent those parameters from exceeding the limits that guarantee a proper level of security (it is usually the case today for existing counters, which most often cannot use the full range of values allowed in a byte but are limited to the first nibble).

2.3.8. Security concerns

The mechanism that is proposed in this invention is designed in order not to lower the security of the smartcard with regard to other kinds of attacks.
It comes with certain prerequisites (which are compulsory for smartcard security anyway).

For example:
1. The counters should be predecremented (as described above), or a flagging mechanism should be put in place in order to prevent tearing attacks and the like
2. When applicable, challenge response should be preferred to credentials comparison
3. In case direct credentials comparison is required (e.g. PIN verification), the credentials bits should be verified in random order, and optionally

should be XORed with random, in order to prevent SPA attacks and the like

4. Credentials should be as unpredictable as possible (this is easily achieved with transport keys which can be obtained by diversification of a random master key for example) and the smartcard OS should enforce that the maximum number of slowed attempts be small enough even for such credentials (e.g. < 256)
5. For credentials that are potentially predictable (e.g. when they are not defined by the system but by the user), a proper security policy should be enforced. For example, the PIN should follow a PIN policy in order to avoid trivial and predictable PIN values, and this can be enforced within the smartcard when possible (e.g. Schlumberger middleware applet does it), in order to prevent PIN guessing attacks
6. Etc.

The third point is important due to the fact that the number of attempts is increased:which means that the likelihood of a power analysis attack success is greater if such a countermeasure is not in place.

The fifth point is important because due to increased number of attempts, the brute force attack (which is ineffective on random credentials) could be replaced by a much more efficient attack in case there are poor PINs (so poor PINs should be rejected, more than ever). Again, the initial value of PINs' second counter shall be much lower than with unpredictable credentials, as stated in 2.3.7.
Provided the basic security guidelines are respected, the security of the smartcard should be improved regarding DOS attacks (the smartcard is no more vulnerable), while not lowered regarding existing attacks.

2.3.9. Optional notification to the external world

Optionally, a command could be implemented on the smartcard in order to notify the external world that a DOS attack (or wrong manipulation) is underway.
The APDU manager could let this command execute without applying the delay loop (the delay would apply to the next command anyway).
Diagnostic tools could poll this command in order to check what's going on. The smartcard would reply with a status word SW_DOS_UNDERWAY or $9000.
Another APDU command would be necessary in order to let the administrators reset the DOS_UNDERWAY flag.

2.3.10. Impact of the optional notification on the external world

The client application does not have to be modified, which is one of the benefits of this invention. Only the administrative tools (CMS, personalization tools, etc) need to be updated, but not the software ("client application") that is rolled out on each end user's PC.

However, in order to be more user friendly, the new behavior of the card could be taken into account in the client application and an explicit warning message could be displayed to the user, thanks to the notification command described in 2.3.9. The client application could then send a "dummy" APDU such as a Select_root that would potentially trigger the waiting loop. If there's indeed a waiting loop, the client application could detect it and notify the user that the card is temporarily unavailable. Otherwise normal processing would proceed.

Without such a modification in the client application, the end user will experience a temporary DOS: the client application will be blocked during the predefined time (for example 30 minutes, as discussed in 2.3.7), which will inform him that something wrong is going on.

In terms of security, it should make no big difference, it would just be less user friendly. After a while, the user would contact some helpdesk, which would quickly diagnose the DOS attack. Since the attack is likely to be rare (provided we implement the described countermeasure), it shouldn't be an issue, and modifying the client application is not necessarily worth the investment. Especially when considering that the virus could circumvent this notification and hide it to the client application and to the end user.

2.4. Alternative possibilities to prevent the Credentials blocking

One can wonder whether there are alternative means to prevent the attack from happening, that wouldn't necessitate any changes in the smartcard. Unfortunately, it seems that such options do not exist.

Some could think of presenting the credentials through a secure channel in order that no forbidden program (virus etc.) tries to verify credentials (with potentially wrong values that could lead to a card blocking). But how do you open the secure channel in the first place? Precisely by authenticating to the card through a ~~successful credentials~~ credentials presentation... So this first step is still subject to the attack. Anyway, secure channel keys are stored in the PC, while by definition the PC is not secure enough and can be compromised by a virus (otherwise we would store all smartcard's secret contents such as RSA private keys in the PC and wouldn't use smartcards as security tokens anymore).

This means that even when secure channels are used for protecting certain credentials presentations, a virus can block the credentials:

- ➢ it can prevent the secure channel from being usable (e.g. by blocking its keys)
- ➢ and/or it can find tricks to use the secure channel illegitimately (e.g. by hacking the secure channel keys in the PC) and block the credentials

2.5. Example of behaviour of an attacked system

Let's consider a typical corporate badge customer, such as an organization with 10,000 employees equipped with PKI badges.
Let's suppose that a virus (e.g. sent to the employees in an e-mail attachment) hits this organization, and that this virus consists in blocking the badges' credentials by presenting wrong values.
Without the invention, the virus could quickly (it's a matter of fractions of seconds) erase the first counter (by a few wrong credentials presentations).
All 10,000 users could be quickly blocked and would have to change their badges. As described in 1.2.2, this could have a very significant financial and security impact on the organization.

Now let's examine what happens when the invention is implemented.
As today, the virus could quickly erase the first counter (by a few wrong credentials presentations).
However, as soon as the second counter starts being decremented, the waiting loop mechanism makes it very long for the virus to erase the counter, and the user is very quickly aware that his badge is being attacked.
Even if the client application does not notify the user that an attack is underway, or if the virus intercepts the notification and prevents the client application from noticing it, the user will experience a temporary DOS.

As stated in 2.3.7, the smartcards could be personalized to wait 30 minutes after each additional wrong attempt, and wait 100 wrong attempts before blocking the credentials.

This means that during thirty minutes, the user will be unable to perform any smartcard-related actions such as

- logon to the PC through Kerberos
- opening a VPN connection with Check Point
- decrypting files on the hard disk using Entrust ICE
- signing e-mails in Outlook

- ➢ unlocking the screen saver with the GINA
- ➢ connecting to a secure web server in SSL through Netscape
- ➢ etc.

To be more accurate, the user can initiate any of these tasks, and in theory it will work but it will take 30 minutes longer than usual (during 30 minutes, the card keeps sending a specific ISO byte, which tells the PCSC stack that it is still processing and that the reader should not time out).

This also means that before blocking the credentials in question, the user should experience 50 hours of DOS per credentials (100 times 30 minutes) without noticing anything abnormal.

Since actually blocking the card usually means blocking a PIN, a PUK and a key (C.F. example in 1.2.1), blocking the card requires above a hundred hours of DOS.

It also requires that ~~the~~ virus is intelligent enough to intercept all legitimate credentials verifications. Otherwise the counters are reset to their maximum value, and the delay (more than 100 hours) restarts from the beginning. Such a virus feature cannot be guaranteed to work in all situations over such a long period (the smartcard could be plugged in another PC that is not infected by the virus, and the card could be unblocked by accident...).

Typical use of corporate badge consists in carrying the badge with you, which means that it is unplugged from the PC as soon as you leave your desk (in order to open the doors, pay the cafeteria, access the parking lots, etc.).

Only when the user is in front of his PC with the smartcard connected can the virus attack the credentials. Let's make the assumption that employees spend an average 5 hours a day in front of their PC (which is a lot, as it's an average for every employees and for every day of work), and that the smartcard is plugged all this time. Even with this pessimistic hypothesis, the virus needs at least one full working month before blocking the card (this corresponds to the shortest possible delay computed in the previous page, which was above a hundred hours).

It is completely unrealistic that users spend more than one month without being able to access any services linked to smartcard (and quite often this includes the inability to use the PC at all, since corporate badges are usually used to login to the PC) without reporting any problem to the helpdesk.
This is extremely unlikely to happen, as it would mean that the users don't depend on their smartcards and don't use it (in which case the attack does not really affect them, and they can most likely wait a few days or weeks before their badge is replaced).
But if users don't actually use their badge, there's no reason for thom to plug it in their PC, which in fact prevents the virus from attacking it (the card needs to be connected in order for the DOS to take effect)...
It is even more unrealistic that all of the 10,000 employees are unable to access the services secured by the smartcard during more than one month and don't report anything.

What will happen is that at least one employee will call the helpdesk, saying that the client application displayed a message such as "your smartcard is under DOS attack, your PC must be infected by a virus, please contact your helpdesk and update your antivirus" or simply complaining that the smartcard does not work.

The helpdesk can analyze the smartcard and verify that there's a DOS attack (thanks to the diagnostic APDU, or just by verifying credentials with a wrong value and checking if it's already in slowed state).

As soon as the helpdesk finds a single user with the problem it could check some other users at random, and if it notices that a few of them are also infected it should apply an emergency plan for the whole organization, for example ask employees to unplug the card from their reader until an antivirus update is available and is successfully run on the PC (and optionally connect to a kind of SSM and perform authentication with all relevant credentials in order to reset all counters to their max value).

30 minutes maximum after the antivirus cleaned the PCs, all badges will be in working order.

If a few users were to be blocked anyway (which is almost impossible), it wouldn't be an issue, their badges could be replaced.
The real issue would be that a high number of users are blocked, and massive quantities of new badges need to be produced and personalized, but this can't happen anymore with the suggested countermeasure.

Anyway, the simple fact that there's a countermeasure should render this attack very unlikely to happen (no more motivation for the attacker).

2.6. Target Applications

The main target for this invention consists of applications built around corporate badges.
However, the invention is potentially applicable to any type of smartcard and for any application.

3. Solution(s) brought by the invention

Massive smartcard destruction by DOS attacks is made much more difficult.

Problems linked to smartcards blocked accidentally by smartcards application developers are significantly reduced.

4. Applicability of the invention

The invention is applicable to any microprocessor smartcard, and potentially to other kinds of hardware tokens.

Claims

1. Smartcard comprising a counter wherein the smartcard also comprises an antenna.

1/1

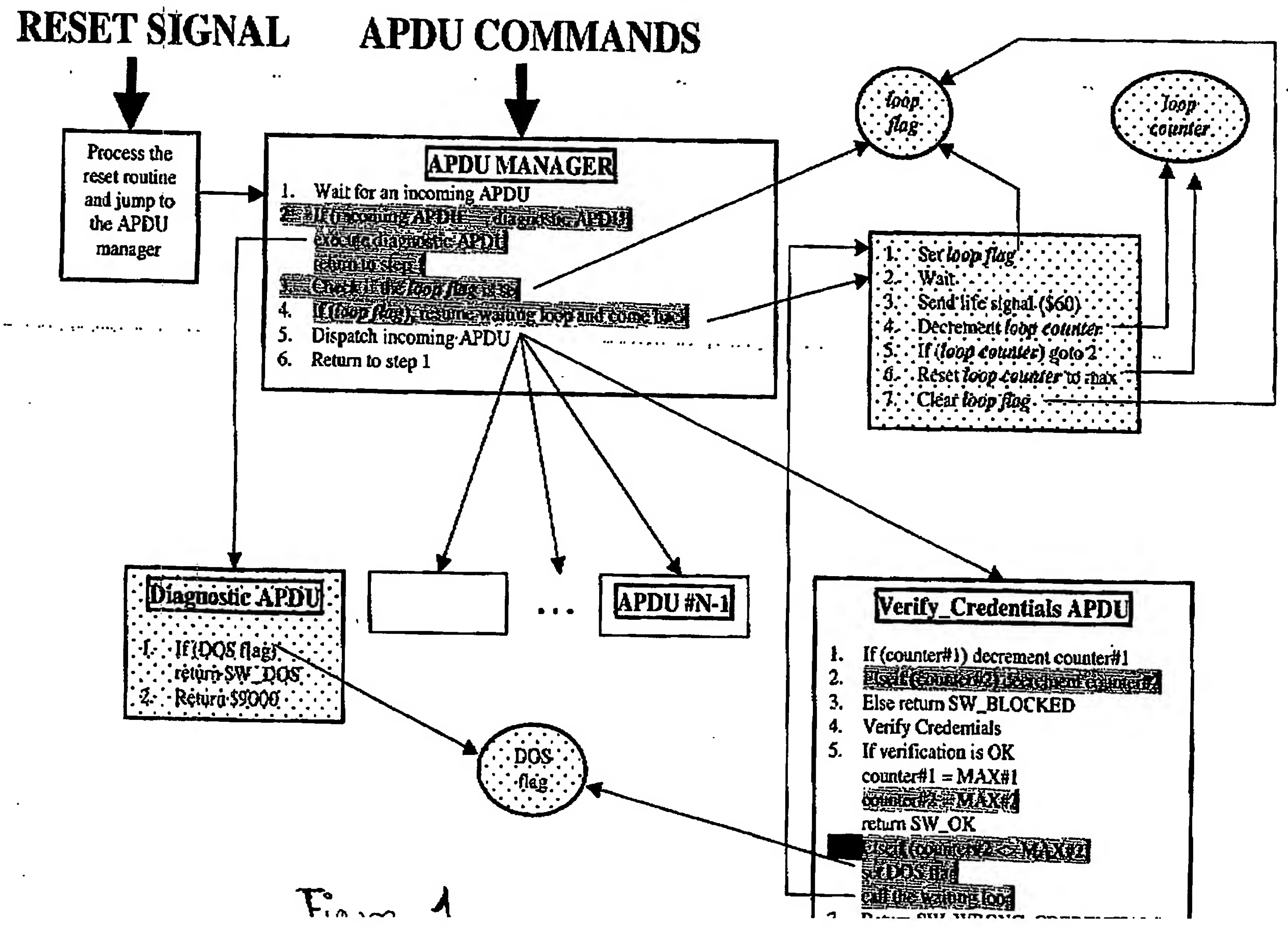
RESET SIGNAL
APDU COMMANDS
Process the reset routine and jump to the APDU manager
APDU MANAGER
1. Wait for an incoming APDU
2. If (incoming APDU == diagnostic APDU)
execute diagnostic APDU
return to step 1
3. Check if the loop flag is set
4. If (loop flag) resume waiting loop and come back
5. Dispatch incoming APDU
6. Return to step 1
loop flag
loop counter
1. Set loop flag
2. Wait
3. Send life signal ($60)
4. Decrement loop counter
5. If (loop counter) goto 2
6. Reset loop counter to max
7. Clear loop flag
Diagnostic APDU
1. If (DOS flag) return SW_DOS
2. Return $9000
...
APDU #N-1
DOS flag
Verify_Credentials APDU
1. If (counter#1) decrement counter#1
2. Else if (counter#2) decrement counter#2
3. Else return SW_BLOCKED
4. Verify Credentials
5. If verification is OK
counter#1 = MAX#1
counter#2 = MAX#2
return SW_OK
Else if (counter#2 <> MAX#2)
set DOS flag
call the waiting loop

Fig. 1